2171 #3

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Art Unit: Unknown Examiner: Unknown

	Arun Gajanan Bahulkar
Case:	P8002
Serial No.:	10/071,981
Filed:	02/05/2002
Subject:	Method and Apparatus for Object-Oriented Access to a Relational Database Management System (RDBMS) Based on Any Arbitrary Predicate

To: The Commissioner of Patents and Trademarks
Washington, D.C. 20231

Dear Sir,

Letter to the Examiner

Enclosed herewith is a certified copy of the provisional application from the Indian Patent Office which is a priority claim for the above-mentioned matter. Also enclosed is a copy of the Information Disclosure Statement that accompanied the US utility patent application submitted to the USPTO on February 5, 2002. Applicant desires that the certified copy of the provisional application be matched with the above referenced case.

Respectfully submitted,

Arun Gajanian Bahulkar et al.

Donald R. Boys
Reg. No. 35,074

Donald R. Boys
Central Coast Patent Agency, Inc.
P.O. Box 187
Aromas, CA 95004
(831) 726-1457

2002

THE PATENTS ACT, 1970

IT IS HEREBY CERTIFIED THAT, the annex is a true copy of Application and Provisional specification filed on 26.7.2001 in respect of Patent Application No.720/Mum/2001 of Tata Consultancy Services(a Division of Tata Sons Limited) of Bombay House, Sir Homi Mody Street, Mumbai-400 023, Maharashtra, India an Indian Company.

This certificate is issued under the powers vested on me under Section 147(1) of the Patents Act, 1970.....

.....Dated this 25th day of January 2002.

(N.K.Garg)
Asst Controller of Patents & Designs

CERTIFIED COPY OF PRIORITY DOCUMENT

FORM-1

THE PATENTS ACT, 1970

(39 OF 1970)

APPLICATION FOR GRANT OF A PATENT

(See sections 5(2), 7, 54 and 135 and rule 33A)

Electronic

1. I/We, TATA CONSULTANCY SERVICES (a Division of TATA SONS LIMITED), of Bombay House, Sir Homi Mody Street, Mumbai 400 023, Maharashtra, India, an Indian Company,

Com 1 pl
Pl accept with prescribed fee
for
26-07-01

2. hereby declare :-

(a) that I/we are in possession is an invention titled OBJECT ORIENTATION

(b) that the Provisional/Complete specification relating to this invention is filed with this application

(c) that there is no lawful ground of objection to the grant of a patent to I/us.

3. further declare that the inventor(s) for the said invention is/are :

(a) ARUN GAJANAN BAHULKAR of Tata Consultancy Services, Hadapsar Industrial Estate, Pune 411 013, Maharashtra, India, an Indian National; and

(b) VILAS PRABHU of Tata Consultancy Services, Hadapsar Industrial Estate, Pune 411 013, Maharashtra, India, an Indian National

ORIGINAL

720/mum/2001
26/7/2001

720 | मुंबई | 2001

26 JUL 2001

Received Rs. [illegible] in Cash
Cash/M.O./P.O. dt. 26/7/01
Vide Entry No. 3120 in the Register of Receipts Number
for
26-07-01
Cashier

4. I/We, claim the priority from the application(s) filed in convention countries, particulars of which are as follows :

(a) [Country]
(b) [Appln.No.]
(c) [Date of Appln.]
(d) [Applicant in Convention Country]
(e) [Title]

N.A.

5. I/We state that the said invention is an improvement in or modification of the invention, the particulars of which are as follows and of which I/We are the applicant/patentee :

(a) Application No.

(b) Date of application N.A.

6. I/We state that the application is divided out of my/our application, the particulars of which are given below and pray that this application deemed to have been filed on date under section 16 of the Act.

(a) Application No.

(b) Date of filing of provisional specification : N.A.

and date of filing of complete specification :

7. That I/We are the assignee or legal representative of the true and first inventors.

8. That my/our address for service in India is as follows : **R.K. DEWAN & COMPANY,** Trade Marks & Patents Attorneys, 78, Podar Chambers, S.A.Brelvi Road, Fort, Mumbai 400 001, Maharashtra, India, Tel. (91) 22-2661662/2663002, Telefascimile number (91) 22-2650159, Email>rkdewan@vsnl.com.

9. Following declaration was given by the inventor(s) or applicant(s) in the convention country :

I/We the true and first inventors for this invention or the applicant(s) in the convention country declare that the applicant(s) herein is/are my/our assignee or legal representative :

[ARUN GAJANAN BAHULKAR] [VILAS PRABHU]

10. That to the best of my/our knowledge, information and belief the fact and matters stated herein are correct and that there is no lawful ground of objection to the grant of patent to me/us on this application

11. Following are the attachment with the application :

(a) Provisional/Complete specification (3 copies)

(b) Drawings (3 Copies)

(c) [illegible]

(d) [illegible]
Copy of General

(e) Power of authority

(f)

(g)

(h)

(i) Fee Rs. 5000/- In cash/cheque/bank draft bearing No.

date on Bank

I/We request that a patent may be granted to me/us for the said invention.

Dated this 24th day of July 2001.

Signature :

Name : (HEMANT DHANSUKHLAL VAKIL)

To :

The Controller of Patents

The Patent Office

at MUMBAI

FORM-2

THE PATENT ACT, 1970

PROVISIONAL

Specification

SECTION – 10

OBJECT ORIENTATION

ORIGINALE
ORIGINAL

TATA CONSULTANCY SERVICES,

(a Division of TATA SONS LIMITED),

of Bombay House, Sir Homi Mody Street, Mumbai 400 023,

Maharashtra, India,

an Indian Company

THE FOLLOWING SPECIFICATION DESCRIBES

THE NATURE OF THIS INVENTION :-

This invention relates to object orientation.

Particularly this invention relates to a method and apparatus for Query abstraction and code generation in object orientation

Object-orientation is becoming the norm in development of software systems. Object orientation has been applied to different architectural layers of a software system successfully. A typical multi-tier client-server software system exhibits the following predominant layers, viz. Presentation layer, Middleware layer, Application layer and Persistence layer.

Object Technology and Relational database management systems have earned their respective places as front-end application development and back-end storage options. Both these methodologies have their own advantages. There are inherent mis-matches between the two technologies, but there are ways and means to overcome the differences.

Organizations have been increasingly turning to Object-orientation to gain the benefits of reusability and faster development. Object technologies enable the rapid development of functionally rich, open and reusable components. As a viable approach to increased heterogeneity, complexity and changing business processes, IS is turning to object-oriented (OO) technology.

Object modeling describes system as built out of objects: programming abstractions that have identity, behavior and state. Objects are an abstraction beyond abstract data types (ADTs), where data and variables are merged into a single unifying concept. Object modeling include

many other concepts: abstraction, similarity, encapsulation, inheritance, modularity, and so on.

There exist a lot of tools and processes, which allow the code generation for simple database accesses based on the primary key, from object model information. There are certain other tools, which allow users to specify a query in SQL (structured query language) and simply wrap this query into a function usable in application processing code.

Object-orientation is therefore successfully applied to the presentation layer and the application layer. Some new component based paradigm have also addressed object orientation in the middleware layer in software systems. But the persistence layer is still not addressed adequately. Object-oriented databases try to address this area, but are immature and have not gained enough acceptability. So the traditional RDBMS are still used in the persistence layer.

E.F. Codd introduced relational model in 1970. Relational database is based on relations i.e. it represents a relational view of data. Tables represent these relations. A row in a table represents a relationship among a set of values. The rows are also called tuples. Relational database systems are based on known theory of Relational Algebra and have strong mathematical foundations. In relational theory, relations define the possible truth statements. The rows or tuples describe the current known truths.

The prior art tools do not provide an object-oriented view of a RDBMS to user code. Neither does it take advantage of specific code patterns that can be specified by user at a modeling level.

The use of RDBMS in the persistence layer also gives rise to so called "impedance mismatch".

Object Relational mis-match: Object modeling describes a system through objects that have identity, behavior, and encapsulated state. Relational models describe a system by information. Objects have a hierarchical relationship whereas relations/tables have a flat structure. Of the primary object-modeling properties, relational modeling seems to have no way of representing any of them directly. Tuples have neither identity nor encapsulation. Tuple attribute values are encapsulated but are pure values, so they have neither identity nor state. This is what is called an impedance-mismatch between object approaches and relational databases.

Fortunately this is not really the case. Predicate logic is quite good at describing the state of the world (or of a model of the world), so relational databases must be quite good at describing the state of an object model. The relational model is expanded to see how easily an object model's state can be modeled.

An object model is based on creating applications composed of objects mirroring business rules and processes. Objects are defined as having both data and behavior. A relational (data) model is based on the storage of data in tables and linking rows between different tables. With an object model, relationships are part of the object definition, while in relational models, relationships are defined by linking rows between tables.

The disparity between the two models gives rise to a mismatch when developers try to link objects to relational data.

There exist tools, which allow a mapping between object model and relational model. Using these mappings, some tools even generate code for simple primary key based accesses to RDBMS in an object-oriented way. But when it comes to access to RDBMS based on arbitrary predicates i.e. non-standard data access methods or NSDM in short, and presenting an object-oriented view of such accesses, there are no tools or methods hitherto available in the prior art.

Object oriented databases have come a long way in the past five years but they are still a relatively new technology. This is one of the reasons why many companies choose not to go with object oriented database technology to solve their database requirements. Also there is a lack of consensus with regard to a formal OO data model, and attempts to characterize an OOD model failed to get universal approval.

Object relational databases have come in the market, which position themselves as the combination of best features from relational and object worlds. ORDBMS have the ability to handle complex information, and have object-oriented facilities and capabilities included. They combine the scalability, reliability, and concurrency of a modern RDBMS with the data modeling features of object-oriented systems. But being a newer technology ORDBMS are not being widely used.

While OODBMS's are built to directly support OT, they are immature in terms of database capabilities and tools. More importantly,

OODBMS's are not widely installed. The vast majority of database management systems in use today are either RDBMS's or legacy databases. It is extremely difficult and time-consuming to integrate legacy relational data structures with object-oriented data structures.

An object of this invention therefore is to provide an object-oriented way to access RDBMS using any arbitrary predicate. An essential property of such an object-oriented access to RDBMS is "polymorphism" which should be guaranteed by the generated code.

This invention therefore discloses a method and apparatus for query abstraction and code generation process that addresses the above issues by exploiting the information captured in the models and SQL (structured query language). The method of this invention requires a generic, extensible, model-repository system that is meta-model driven.

The method and apparatus described herein is model based generative method.

In particular the method of this invention assumes that an Object-Relational mapping can be specified in such a repository and can be used by query-abstraction. That is, query abstraction does not define its own mapping, but uses a mapping already in place. The current query abstraction defined herein assumes a particular object-relational mapping viz. One to one class to table mapping with replication in subclasses. However the query abstraction in accordance with the method and the apparatus of this invention in no way precludes the usage of other types of object-relational mapping.

In a relational database, the schema is made up of tables, consisting of rows and columns, where each column has a name and a simple data type. In an object model, the equivalent to a table is a class (or type), that has a set of attributes (properties or data members). Object classes describe behavior with methods (member functions).

In a relational database, a tuple (row) contains data for a single entity that correlates to an object (instance of a class) in an object-oriented system. In addition, a stored procedure in a relational database may correlate to a free function in an object-oriented architecture.

Therefore, the mappings essential to object/relational integration are between a table and a type, between columns and attributes, and between a row and an object, and between a stored procedure and a free function.

Table	⇔	Class (type)
Columns	⇔	Set of attributes of the class.
A tuple (single row)	⇔	Object (class instance)
A Stored procedure	⇔	free function

An example of a class to table mapping is shown in Figure 4 of the accompanying drawings.

Associations

Relationships between objects map to foreign keys between rows. How each mapping is implemented, however, has a significant impact on the performance and flexibility of the resulting application. There are several choices for mapping object relationships to relational tables:

Embedded foreign key: This is the most common approach for one-one and one-many relationships. In this case, for a given class the primary key of a related class is embedded in the class itself. This results in a performance characteristic better than the "distinct table".

Distinct table: In this approach the relationship is represented as a distinct table in the database. This approach provides the most modeling flexibility since it makes adding and removing relationships transparent to other tables. However, this approach can be expensive if the relationship is frequently traversed.

In general, the embedded key approach is most efficient for one-to-one relationships and one-to-many relationships, while the distinct table approach is required for many-to-many relationships.

A typical meta-model for object relational mapping and query abstraction is now hereinafter described with reference to Figure 1 of the accompanying drawings.

The model has meta objects Class, Attribute, Domain, from the Object domain and meta objects Table and Key from the RDBMS domain. The meta object, Query is used to model arbitrary-predicate selections from the RDBMS.

MetaAssociations between MetaObjects such as, Class MapsTo Table, Class Has ClassAttribute, Table Column ClassAttribute, Table Has Key, Key ComposedOf ClassAttribute, ClassAttribute Has Domain, Key RefersTo Key, Domain ComposedOf Domain, define the mapping of the Object Model to RDBMS Schema.

The MetaObjects in the Model have MetaProperties, which represent the generation specific database design attributes. Meta properties that are used by the query abstraction are given in figure 2a of the accompanying drawings with their default values.

The query meta object has the following meta properties:

- QueryName
- QueryText
- QueryType
  - C – For Cursor methods
  - S – For Single row selects
  - P – For paging methods
  - CP – For Cursor and Paging methods
- DeleteMethodReqd
- UpdateMethodReqd

The QueryName becomes the name of the class in which all the methods of the query are present. The user writes the actual SQL in the QueryText. QueryType is used to generate different methods based on the selection. ***Cursor methods*** are used to select multiple rows and process them one after another till all the records are exhausted. ***Paging methods*** are typically used in User Interfaces where the user will be shown a small set of records and depending on the user's selection, the next set of records or the previous set of records are shown. This action helps in reducing the traffic from the server to the client since only the minimal set of records which can be shown in the screen are fetched from the database. ***Single Row Get*** is used when aggregate functions like SUM, COUNT are used in the SQL.

The following associations are needed by the query meta object:

- Query **Returns** Class (M:1)

  This association represents the output of the query.
- Query **HasInput** Parameter (1:M)

  This association represents the input for the query.
- Parameter **OfType** Class (M:1)
- Parameter **ValidatedBy** Domain (M:1)

The last two associations define types for input and output of the query. These parameters help in providing type information.

The query class is used when user wants to use a non-primary key based query. Paging, cursor access and complex select clauses resulting in a single row output are allowed. A cursor type query is marked to generate Update and/or Delete methods, which generates the positioned update and/or delete functions. The QueryText meta property contains the text of the query. The query text is entered as SELECT statement with an INTO clause, other clauses being optional. The syntax used is similar to standard SQL syntax, only difference being the way host variables are specified. Users enter the input and output parameters for the query. The users define types for the parameters, and only these parameters are used in the query text.

For query implementation a query is specified in the model as per the meta-model specified above. The query manifests itself as a class in generated code. Based on information present in model, various methods are generated for this class, as explained below. The user, can instantiate an object of this class and uses it in an object oriented way in his code.

An example of a Query is shown in Figure 2 of the drawings and is described herein below.

The Query is modeled as ...

```
emplist Returns ContrEmployee
emplist HasInput p_date, p_status
p_date ValidatedBy d_date (which is of RDBMS type Date)
p_status ValidatedBy d_int (which is of RDBMS type  Number)
```

Query Text should be entered as an embedded SELECT statement. An example QueryText is as follows.

Query Text for Query "emplist"

```
SELECT EmpId, EmpName, EmpDept
INTO :3.ContrEmployee.EmpId,
      :3.ContrEmployee.EmpName,
      :3.ContrEmployee.EmpDeptId
FROM      ContrEmp,
      Project
WHERE ContrEndDate - TO_DATE( :1.p_date, 'YYYYMMDD') < 10
  AND EmpProject = ProjectId
  AND ProjStatus = :2.p_status
```

Options in generation of functions

If,

- QueryType is marked 'S', Get () function is generated. (To be used only if the output expected out of the query is a single row.)
- QueryType is marked 'C' for Cursor operations. Open (), Fetch () and Close () functions are generated.
- QueryType is marked 'P', GetM(), GetM_Fwd() and GetM_Bwd() functions are generated.
- UpdateMethodReqd is marked 'Y', then UpdateCursor () function is generated.
- DeleteMethodReqd is marked 'Y', then DeleteCursor () function is generated.

UpdateMethodReqd and DeleteMethodReqd should be marked 'Y' only if the QueryType is Cursor. In both these cases, the Open () function will open the cursor for '*update*'.

Host variable representation

Host variables in SQL text should be represented as follows.

Host_var :: =<N>.<class/domain name>.<attribute name>

In case of simple domains, the attribute name is not required.

N is the serial number of parameter, the parameter numbering starts with input parameter. The first parameter being 1 and so on. The output parameter gets the last serial number. Some examples of host variable are

```
:1.Employee.EmpId
:2.Address.Line1
:3.d_Date
```

The parameter sequence is determined by ordering of association (HasInput) in the model. If parameter to query association is re-sequenced in the model, then query needs to be altered. i.e. the sequence numbers of the parameters need to be changed in accordance with the model.

Methods for retrieval

The following methods can be generated for a query. But QueryType 'S' and 'C' are mutually exclusive and hence only one set of functions will be generated.

- **Get (parameter1, parameter2, ..., object_var) : Status**

Get() method is generated when the QueryType in the model is set to S. This method takes as many parameters as defined in query. All of these parameters are input parameters. The object_var is the object which the get() would return. The method always returns a single object as output. The return value of the function can be a status for quick error checking.

- **Open (parameter1, parameter2, ..., object_var) : Status**

  Open, Fetch and Close are generated if the QueryType is set to C. This method takes as many parameters as defined in query. All of these parameters are input parameters. The object_var is the object which the fetch() would return. The method opens a cursor to facilitate fetching of objects from the database, using the specified query. The return value of the function can be a status for quick error checking.

  This method should be used to retrieve objects satisfying criteria specified in the query. This method would facilitate fetching of objects one by one. Fetch function (described below) should be used for this.

- **Fetch (parameter1, parameter2, ..., object_var) : Status**

  This method takes different set of input parameters based on the query. The object to be retrieved as specified in the query is the object_var, which is the output. The return value of the function can be a status for quick error checking.

  This method fetches objects satisfying criteria specified in the query. It expects a call to open method is made before this method is called.

- **Close (parameter1, ..., object_var) : Status**

  This method takes only Error flag as an output parameter. It closes the cursor opened for this query. It returns errors and any related information in Error object, which is an output parameter to it. The return value of the function can be a status for quick error checking.

  This method should be called corresponding to each Open cursor method call.

1.1.1 Methods for modifications

- **UpdateCursor (parameter1, parameter2,... object_var) : Status**

  This method is generated if UpdateMethodReqd property of the query is marked 'Y'. This method takes object variable (object_var) as input parameter. It updates the object (supplied as object_var) in database. The return value of the function can be a status for quick error checking.

  This method expects call to open and Fetch methods are done before calling this method. It would update the last fetched object in database.

- **DeleteCursor (parameter1, parameter2,..., object_var) : Status**

  This method is generated if DeleteMethodReqd property of the query in marked 'Y'. It deletes the last fetched object, from database. The return value of the function can be a status for quick error checking.
  This method requires call to Open and fetch methods before it is called. It would delete the last fetched object, from database.

1.1.2 Paging Methods

These methods required for doing paging on a client user interface are generated when the QueryType is set to P or CP. These methods are to be called by the application in proper sequence to get the desired effect of scrolling just by fetching the optimum number of records that could fit in the window of the GUI.

- **GetM (parameter1, parameter2,..., DynamicArray *object_var) : Status**

  This method is called as the first step to show a set of records in a scrollable window of the GUI. The number of records to be selected from the database is decided based on the GUI design and is defined.

GetM () gets one extra row along with the required number of records to be displayed in the screen. The subsequent calls are based on the user action.

- **GetM_Fwd (parameter1, parameter2,..,ContObject, DynamicArray *object_var) : Status**

  This method is called to get the pre-defined set of records, which follows the logical sequence as defined in the query from the one that is given in the ContObject or the Continuation Object. If the user scrolls down, then the application should call this method, which gets the next set of records in the forward direction.

- **GetM_Bwd (parameter1, parameter2,...,ContObject, DynamicArray *object_var) : Status**

  If the user scrolls upward, then a call to GetM_Bwd is done by the application, which gets the set of records in the reverse direction as present in the continuation object.

Other usage of query abstraction

The query abstraction can be used to automatically generated methods for accessing associated objects. The UML models allow specification of association between classes in model. It is desirable that user be provided methods to access these associated classes through methods. Accessing such associated class normally involves database join. Such association accessing methods can be modeled in terms of above meta-model and a corresponding SQL text can be generated from model information. With these specifications, one can generate the access methods the same way methods are generated for individual query.

A relational database, typically do not handle inheritance. But if an object model is mapped to a relational schema, then a way should be found to represent inheritance in the relational database.

There are four possible strategies used for this mapping.

Horizontal Partitioning: In this case, only leaf classes are mapped to tables and include all of their inherited attributes. This approach may give improved performance since only one table needs to be accessed for instances of a given leaf class.

Typed Partitioning: Another way to handle inheritance is to map all classes in an inheritance tree to a single table, using a type column to distinguish between subclasses. This enables the retrieval of objects from multiple classes in a single query, but violates normalization.

Vertical Partitioning: In this type of mapping parent classes in the object model class maps to a corresponding table.

Replication Partitioning – Every derived class will duplicate the base class' attributes.

Replication: All the attributes of the parent class(es) are replicated in the derived class' table, with separate table for each class in the hierarchy.

The Advantages of the using the method of this invention are as under

1. Any arbitrary predicate RDBMS access can be made in an object oriented way because of query class.
2. The methods generated support polymorphism, that is any instance of the type (same type or derived type) modeled as input/output parameter can be supplied to generated queries.
3. The code pattern for various methods is automatically generated based on model information, user does not have to code for it.
4. All the complex database accesses are available in model, hence it's easier to do performance-related analysis, such as access path analysis.

Although the invention has been described in terms of particular embodiments and applications, one of ordinary skill in the art, in light of this teaching, can generate additional embodiments and modifications without departing from the spirit of or exceeding the scope of the invention. Accordingly, it is to be understood that the drawings and descriptions herein are proffered by way of example to facilitate comprehension of the invention and should not be construed to limit the scope thereof.

Dated this 24th day of July 2001

Mohan Dewan
Of R. K. Dewan & Co.,
Applicant's Patent Attorney

TATA CONSULTANCY SERVICES
ME (TATA SONS LIMITED)
: / MUM / 01

NO. OF SHEETS : 4
SHEET NO. : 1

PROVISIONAL SPECIFICATION

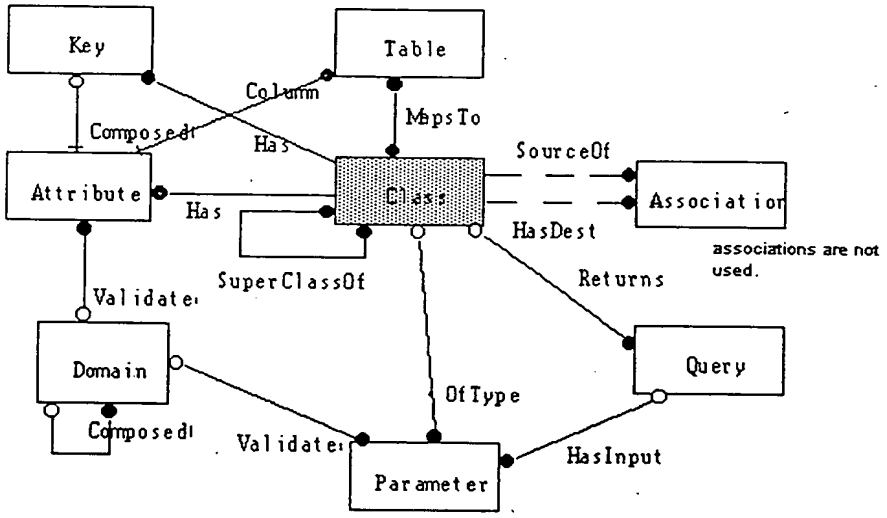

Figure - 1
Meta model for object relational mapping & query abstraction

DR. MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

2 JUE 2001

TATA CONSULTANCY SERVICES

NAME :(TATA SONS LIMITED) NO. OF SHEETS : 4

NO. : / MUM / 01 SHEET NO. : 2

PROVISIONAL SPECIFICATION

MetaObject	MetaProperty	Description	Values	Default Value
Class	PERSISTENCE_TYPE	Persistent or Transient Class	P/T	T
	VERSION_FLAG	Soft Locking required	Y/N	N
Table				
Attribute	Optionality	Optional or Mandatory (NULL or NOT NULL)	O/M	M
Query				
	QueryType	SimpleSelect/Cursor	C : CursorSelect CP : Cursor&Paging MC: MultirowCursor P : Paging PMC : Paging and multirow cursor S : SimpleSelect	-
	UpdateMethodReqd	Update Method required?	Y/N	N
	DeleteMethodReqd	Delete Method required?	Y/N	N
Parameter				
Domain	DataType	Database Data Type	CHAR/Date/ NUMBER/ VARCHAR2/ LONGRAW	-
	CompoundFlag	Domain composed of Other domains OR Simple Domain	Y/N	-
Key	KeyType	Primary/Alternate Key	P/A	-

Figure - 1a

DR. MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

25 JUL 2001

PROVISIONAL SPECIFICATION

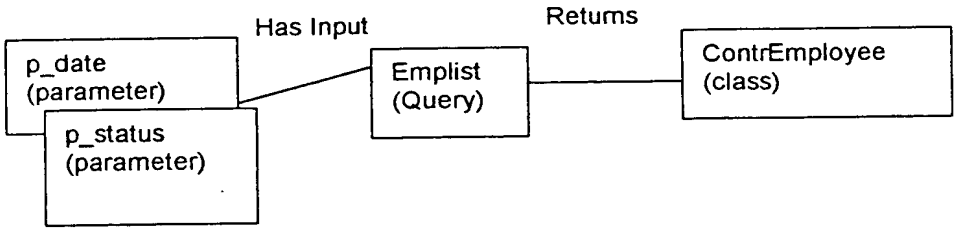

Figure - 2 User model of a query

```
...
emplist Returns ContrEmployee
emplist HasInput p_date, p_status
p_date ValidatedBy d_date (which is of RDBMS type Date)
p_status ValidatedBy d_int (which is of RDBMS type  Number)
```

Figure - 2a

DR. MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

26 JUL 2001

PROVISIONAL SPECIFICATION

```
SELECT EmpId, EmpName, EmpDept
INTO :3.ContrEmployee.EmpId,
     :3.ContrEmployee.EmpName,
     :3.ContrEmployee.EmpDeptId
FROM  ContrEmp,
      Project
WHERE ContrEndDate - TO_DATE( :1.p_date,'YYYYMMDD') < 10
  AND EmpProject = ProjectId
  AND ProjStatus = :2.p_status;
```

Figure - 2 b

Host_var :: = <N>.<class/domain name>.<attribute name>

Figure - 2 c

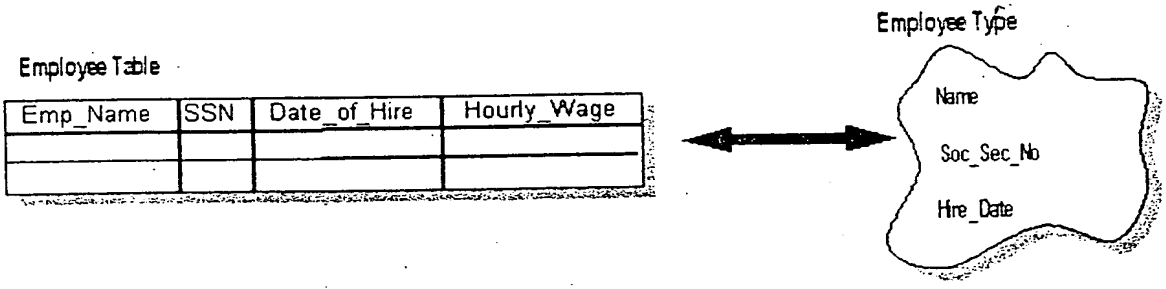

Figure - 3 Class to table mapping

DR. MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

26 JUL 2001